大規模リレーションにおけるデータ縮退を用いたグラフ算法

3N-6

津高 新一郎　大森 匡　田中 英彦
東京大学 工学部

1 はじめに

データベースの分野において近年特に注目を集めているテーマとして、再帰問合せ(Recursive Query)がある。これは自分自身を再帰的に呼び出してデータベースを操作する問合せであり、その代表的な例として、親子関係が与えられた際にそこから先祖関係を求める問合せがある。とりわけ制約演算子つきの問合せ、すなわちある特定のデータに関する再帰的問合せはもっとも有用な問題としてマジック集合法など多くの算法が提案されてきた。

再帰問合せはデータベースリレーションをグラフとみなした上でのグラフ算法で実行できるものが多い。例えば[1]推移閉包演算、[2]単一出発点最短経路演算、等はいずれも再帰問合せの処理に必要な演算である。マジック集合の計算は上の[2]のカテゴリに入る。本稿ではデータベースリレーションの単一出発点最短経路問題を、共有型データベースマシン上で実行する算法について提案する。

2 接続情報への縮退と結合演算による復元操作

大量のデータを扱うグラフ算法には、つぎのような問題点がある。

- 一般にグラフ算法にはマークをつけながらグラフ内をたどるものが多く、算法自体の並列性は限定されていることが多い。
- 算法実行中に必要な中間データが多いためにそれらのデータのバッファリングが全体の実行速度を支配する。そのため各算法ごとにバッファリングを工夫しなければ全体的なパフォーマンスは上がらない。

ここで、我々は、

- 通常グラフ算法で直接必要とされる情報は1タプル当たり数バイトに縮退できることが多く、タプルの残りの情報は単に計算結果に付随する情報として利用されるにすぎない。
- コストのうちの大部分を入出力コストが占め、グラフ算法の計算そのものによるコストの割合は比較的少ない。

という二点に着目し、次のような3フェイズからなる手法を提案する。

縮退フェイズ ディスク上のリレーションからグラフ算法で直接必要とされる情報のみを取り出し、さらにデータの圧縮を行うことにより1タプル当たり数バイトのデータとする（以後接続情報と呼ぶ）。

演算フェイズ 接続情報を用いてなるべくメインメモリ内で計算を行う。もしメインメモリからあふれた場合は中間リレーションとしてディスクに書き出して管理する。この演算フェイズは前の縮退フェイズとあわせてパイプライン的に実行される。

復元フェイズ 演算結果として得られた接続情報と元のリレーションに対して結合演算を施し、答となるタプル集合を復元して結果リレーションとしてディスクに書き出す。

この手法の利点には次のようなものがある。

- 予めグラフ算法で用いるデータを一桁以上小さくしておくことによって、グラフ算法実行中に生じるメインメモリからディスクへのデータのあふれを最小限に抑え、入出力コストを極めて小さくすることができる。
- 結合演算による復元操作はディスク内の連続領域アクセスによるのでデータベースマシンが得意とする演算の一つであり、コストとしては安い。そのうえディスクモジュールの並列化による台数効果もある程度望める。

3 単一出発点最短経路問題

最短経路問題を解くとは、辺に関する情報を持つタプルの全集合から、最短経路を構成する辺の情報を持つタプルを全て求めることである。この時用いるアルゴリズムとして、ダイクストラの算法を用いるが、この算法には次の二つの集合が必要となる。

Graph Algorithms On Large Relation Using Data Reduction
Shinichiro TSUDAKA,Tadasi OHMORI,Hidehiko TANAKA
Tokyo University

解答集合 最短経路が既に計算された頂点、及び最短経路をなす辺の集合

中間集合 解答集合から直接のびている辺及びその先の頂点のうち、現在のところその頂点への最短経路をなすものの集合

解答集合は計算の進行と共に増加し、最終的には最短距離で到達可能な頂点の集合となる。一方中間集合は計算開始時、終了時は空集合で、計算中に最大となる。例えば、各頂点から出ている辺の数を8とし、辺の終点はランダムであるようなグラフ（これを次数8のランダムグラフという）を考えると、中間集合の大きさは計算の進行に伴い図1.のように変化する。この中間集合の占めるメモリ領域をできるだけ少なくすることがコストの減少につながる。

また、ダイクストラの算法は、中間集合、解答集合を交互に計算していくことによって逐次的に進行する。そのため並列化による高速実行は困難である。

4 結果

表1.のような環境下でシミュレーションを行った結果を図2.に示す。例題としては、図1.と同じく次数8のランダムグラフを用いた。中間的なメモリサイズのときは1桁以上の性能向上が見られる。また、ディスクモジュールの並列化による高速化も若干見ることができる。

5 おわりに

巨大なデータベースリレーションのグラフ算法は、中間データが大きいためバッファリングが難しく、また並列化による性能向上が困難であるという欠点を持っていた。このような算法では、グラフの接続を表す情報をいかに小さくおさえるかというデータ縮退の戦略、及びそれを可能にするようなファイル編成法が非常に有効かつ重要であることがシミュレーションによって明らかとなった。

参考文献

[1] F.Bancilhon and R.Ramakrishnan."An Amateur's Introduction to Recursive Query Processing Strategies" ,ACM-SIGMOD Int'l Conf.on Management of Data '86,pages 16-52,1986.

[2] Hopcroft J.E.Aho,A.V.and J.D.Ullman. "The Design and Analysis of Computer Algorithms", Addison Wesley,1974.

表1: 測定環境におけるパラメータ

全タプル数	10k 件
解答となるタプル数	1k 件
1タプルのバイト数	128 bytes
縮退情報のバイト数	12 bytes
1ページのバイト数	512 bytes
1ページのタプル数	4 個
メインメモリのバイト数	16k ～ 64k bytes
メインメモリのページ数	32 ～ 128 pages
1ページアクセス時間	25 ms

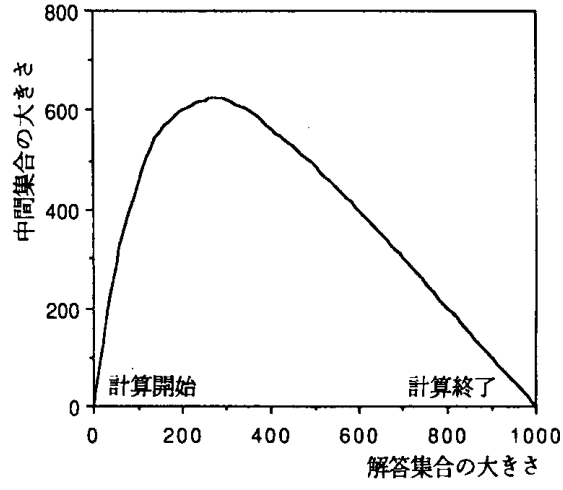

図 1: 中間集合と解答集合の大きさの変化

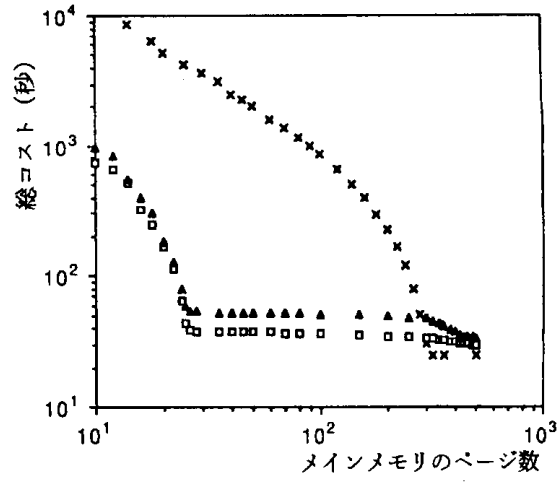

図 2: 計算結果

- ✖ 初歩的な方法（ディスク1台）
- ▲ 提案する方法（ディスク1台）
- □ 提案する方法（ディスク10台）